AMATH 482 Homework 2

Gonzalo Ferrandez Quinto

February 8, 2020

1 Introduction and Overview

On this writing, we are trying to analyze the songs of *Sweet Child O' Mine* by Guns N' Roses and *Comfortably Numb* by Pink Floyd. In both cases a guitar riff is played with the complication that on Pink Floyd's, the bass and beat are kept in the background. the goals of this papers are three:

1. Create a music score for the guitar in the GNR song and a music score for the bass in *Comfortably Numb* song
2. Isolate the bass in *Comfortably Numb* and create an audio file of it
3. Isolate the guitar in *Comfortably Numb* and create a music score of it

When creating a music score, what we are really plotting is the frequency of the note we want to play and at what time. Therefore not just a Fourier transform will be needed as we will also need information of when was this note played. To do these tasks I will be implementing the Gabor transform and other filtering methods. In addition, one of the main complications are overtones which will be explained in the theoretical background.

2 Theoretical Background

2.1 The Gabor Transform

2.1.1 definition

To accomplish my objectives, I will be using the Gabor Transform. The Gabor Transfrom can be understood as a Fourier Transform which has been taken for a signal around some specified time. To be able to select the time, a Gaussian window function is used, centered at some specific time. Therefore being able to depict what are the frequencies (notes) that being played around that instant.

$$\tilde{f}_g(\tau, k) = \int_{-\infty}^{\infty} f(x) g(t-\tau) e^{-ikt} dt \tag{1}$$

Here $g(t-\tau)$ represents the Gaussian Function with variable center τ which is sliding across the time domain of the function, selecting the time interval where we are localizing the Fourier Transform. Here the Gaussian function (window function), is selecting the values around its center while is canceling the rest as they get further away. We are able to determine what range of values get canceled by varying the width(standard deviation) of the fuction. It should be noted that now the resultant function is $\tilde{f}_g(\tau, k)$, which depends on two variables. Therefore giving the frequency spectrum (k) at some specific time.

As I said before, usually a Gaussian function is used, but ant function that is real and symmetric can be used. In addition, it must satisfy that:

$$\|(g)\|_2 = \left(\int_{-\infty}^{\infty} |g(t)|^2 dt \right)^{\frac{1}{2}} = 1 \tag{2}$$

2.1.2 Heisenberg's Uncertainty Principle

The Gabor Transform is by definition picking a window of the original function and calculating what are the frequencies present in that segment. This battles directly with the ides of Heisenberg's Uncertainty Principle. Therefore if the time window is narrowed too much we will have a flat signal in frequency space. While if the time window is really big, the Transform's output will be the original Fourier transform, therefore giving all the frequencies that are present but with no time reference. This means that the window function needs to be chosen at some value for which gives some error both in time and frequency domain but that is able to give us information for both.

2.1.3 Discrete Gabor Transform

All the results explained above are only applicable to continuous infinite functions as the integral is hinting. For the analysis of the music, we will be needing a discrete version.

$$\tilde{f}_g(m,n) = \int_{-\infty}^{\infty} f(t)g(t-nt_0)e^{-2\pi i m w_0 t}dt \tag{3}$$

Where now $\tau = nt_0$ and $k = mw_0$ therefore having discrete values at which you are able to center the window function and also having discrete frequencies. Although here, the input still is a continuous function taken under an integral. Therefore we must use the form in which only a discrete amplitude-spatial values are the input, the same way in which a song would be sampled.

$$\hat{x}_{n,m} = \frac{1}{N}\sum_{j=0}^{N-1} e^{\frac{w_0 j 2\pi}{N}} x_j g_{jm} \tag{4}$$

Here given the input of a row vector x_n which are the sampled amplitudes in the time domain and a matrix g with the same number of rows. The columns of g correspond to the Gaussian contribution coefficient of a point in time given some mt_0 value. In addition, N represents the number of the equally spaced time samples in the input vector. Each one of the entries can be calculated by taking the calculation $g_{im} = e^{-a(t_i - mt_0)^2}$ where the vector t represents your time vector and mt_0 your discrete time step. The term a in the exponential will be inversely proportional to the width of the Gaussian function.

In addition, this approach is implementing the Discrete Fourier Transform therefore it will imply that aliasing is present (finite number of frequencies). At the same time that the input vector is assumed to be 2π periodic, therefore needing to be scale the frequencies by $\frac{2\pi}{L}$. The finite number of frequencies that will be found are k = [-N/2,-N/2+1,...,-1,0,1,2,...N/2-1]

2.2 Overtones

When a string instrument is being played, standing waves are being created on the cords. These are vibrations which are contained within its boundaries and when perturbed produce nodes and anti-nodes in the cord. A fundamental frequency will be the one that given the length of the cord, only contains one anti-node and two nodes on the sides. As in wave propagation $v = f\lambda$, where v will only depend on the linear density of the string. Therefore for any given string, by reducing its length by pressing with the finger we will be producing a bigger frequency sound. For some given cord length, apart from the fundamental, frequency there will be overtones which are all the multiple integers of the fundamental frequency that will produce a wave with a wavelength capable of fitting in the cord. These produced overtones will be playing the same note but with a higher pitch. Therefore when someone is playing, apart from the fundamental frequencies, one will also be creating by consequence its overtones. This is the main problem that I will encounter when analyzing the time-frequency spectrum of Pink Floyd song. As the bass is capable of producing overtones that locate themselves in the frequency range of the guitar which interfere with the creation of the music score.

I will take the bass frequency range to go from 0-300 Hz where the fundamental frequencies locate themselves. For the guitar I will consider the 300-800 Hz range. In addition it is worth noting that overtones from a given instrument produce a lower amplitude sound which will come handy when cleaning the histograms.

3 Algorithm Implementation and Development

3.0.1 Music Score

To achieve the first objective of creating the Music Score for the Guitar in GNR and the Bass in Pink Floyd's I had to do the following steps :

1. Create the Amplitude-time vector from the music file
2. Multiply the amplitude vector by a time sliding window function
3. Take the Fast Fourier Transform
4. Store this values in a matrix and display them using a spectogram

To transform the music file to a vector I used the function `audioplayer()` which outputs a vector with the amplitudes at discrete equally spaced times and the sampling frequency, which is the number of discrete times measured per second. After this inside a for loop, I created A Gaussian coefficient vector for each one of the τ values. For this part I used a time step value of $t_0 = 0.1$ and a Gaussian width value $a = 500$ I multiplied this Gaussian vector with the amplitude vector and then took `fft()` (Fast Fourier transform of it). This vector now represents all the frequencies present around that time step. To make the spectogram look better afterwards, centered it around the maximum frequency amplitude in the range from [300-800]Hz for the base and [0-300]Hz for the base as here is where the fundamental frequencies are present. This will filter out overtones at each time step as these will have a smaller contribution if the frequency domain. After this I took the `fftshift()` of each vector and stored it as a column in a matrix.

After this I created the spectogram using the `pcolor()` command. This receives a matrix as an input along with the x-y axis values. In this case the matrix input was the frequency-time matrix created before. Where a higher magnitude in an cell corresponds to a bigger presence of that amplitude at that time. I did this for both cases of the Guitar in GNR and the bass in Pink Floyd. In addition, I used the command `set()` which zooms in the desired freqeuncies.

I did one extra step for the case for the bass as here I also implemented a filter function to clean up the histogram. Basically for each time step tau, I filtered around the frequency with highest amplitude in the range of 0-300Hz using the `max()` command. Therefore seeing more clearly the notes that are being played. For this Gaussian filter I used width of 0.2.

3.0.2 Filtering bass in Comfortably Numb

To filter out everything except the bass for task number 2 I used a low-pass filter in frequency space. This was really simple as we know that no overtones from the Guitar can be in the range of the bass. Therefore for this case I did not need to implement the Gabor Transform but only the **fft()**. Therefore after converting the audio file into a vector using `audioplayer()`, I Fast Fourier Transformed the amplitude vector for Comfortably numb into Frequency space. Then I made zero all the contributions of frequencies that are not in the [0-300]Hz range. Afterwards, performed the **ifft()** to put it back in time domain where now all the frequencies above the bass fundamental range are trimmed.
For isolating the Guitar I had to do a more complicated task as here there are overtones coming from the bass. I did this in the 0-10 second interval. The first thing I did was as before put the music in vector form using `audioplayer()` and use the Gabor transform in the same way. Afterwards, I used two different times of filters. First I targeted the overtones of each one of the notes being played by the bass. I just targeted the 3 overtones above the one that was being played. Therefore making zero multiples of the frequencies that are being played by the bass in figure 2. I did this by storing each one of the maximum frequencies being

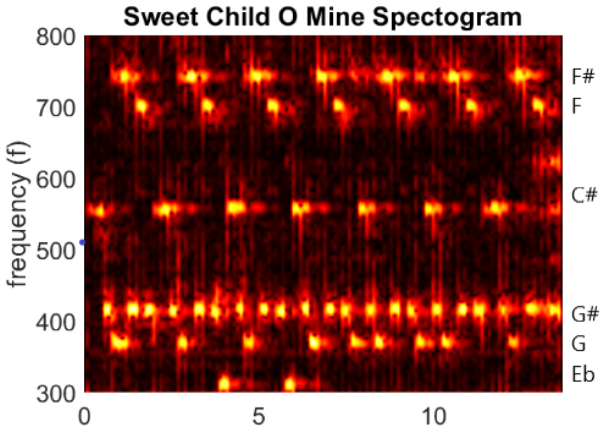

Figure 1: Spectrogram(Music Score) of the Guitar in the GNR song

played at each time interval. The second filter that I implemented was a hard filter that targeted frequencies in the [0,300] range (cancelling the fundamental bass frequencies) and also in the [2000,2200] interval. As in the second one some background sound was found.

4 Computational Results

I was first able to create the spectograms for both the guitar and the bass in the different songs. I did not plot all the 60 seconds for the bass as it would have been too many graphs. We can see for example the riff for GNR how the only notes being used are six and are being over time. Only two notes, where instead a Eb is also played. For the bass music score, it is clear that fewer notes are being played. Over the 0-10 second interval long notes are played at frequencies of B and A. Later in the 0-30 interval, a note G is also played.

In the second part of the task I had to isolate the music of the bass in comfortably numb. This was an easy task as only filtering in the Frequency domain was needed. Here I filtered out all frequencies over 300Hz with a square filter. In my opinion, the results are great as one can clearly hear the bass playing. To reproduce the result, one can use the code in Appendix B Listing 3.

For task three I was not able to return good results in a spectogram plot as it was not appreciable the notes that are being played. In contrast, when performing the inverse Gabor transform to return the music into time domain, I was actually able to only here the guitar being played although with a high amount of static sound, that may account for the error in the spectogram.

5 Summary and Conclusions

During this whole project, the value of Fourier Transform has been valued in applications. It is incredible that one can produce a music score from the music file using a Gabor Transform. One of the principal remarks is that the value of the width of the window function had a lot of importance in the quality of the final plots.

Appendix A MATLAB Functions

Add your important MATLAB functions here with a brief implementation explanation. This is how to make an **unordered** list:

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `fftn()` is the N-dimensional analogous function of the `fft()`

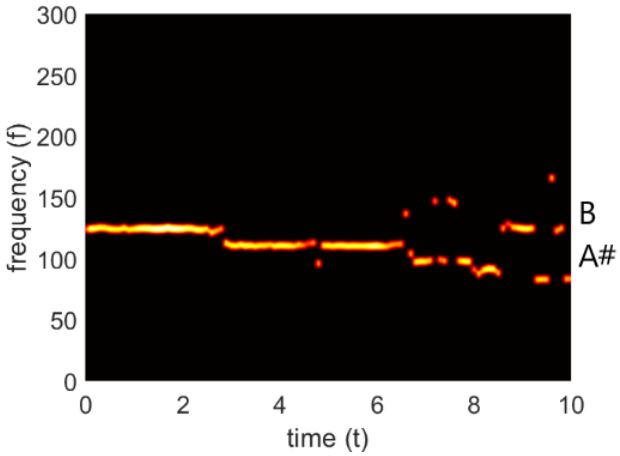

Figure 2: Spectrogram(Music Score) of the bass in the Pink Floyd song for 0-10 second interval

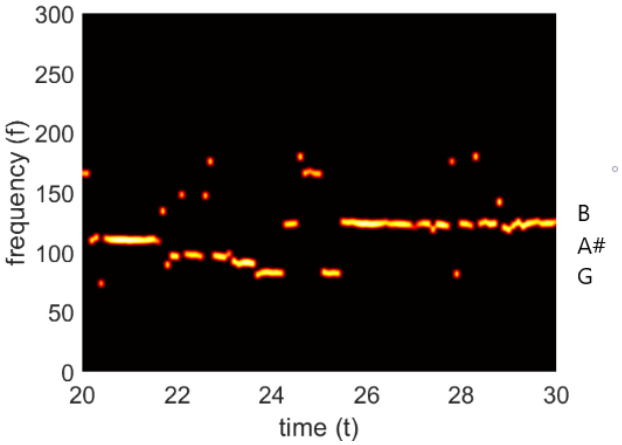

Figure 3: Spectrogram(Music Score) of the bass in Pink Floyd song for 20-30 second interval

- `ifftn()` is the IFFT for N-dimensions
- `max()` returns the maximum value of an array
- `pcolor()` produces a color map given a matrix and the axis
- `audioplayer()` given a music file produces the amplitude-time vector and the sampling frequency
- `set()` zooms in a region of the spectogram

Appendix B MATLAB Code

```
% Clean workspace
figure(1)
[y, Fs] = audioread('GNR.m4a');
trgnr = length(y)/Fs; % record time in seconds
plot((1:length(y))/Fs,y);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Sweet Child O Mine');
p8 = audioplayer(y,Fs);
S = y';

%% Creating a spectrogram
L = length(S)/Fs ; n = length(S);
t2 = linspace(0,L,n+1); t = t2(1:n);
a = 500;
tau = 0:0.1:L;
k = (1/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);
Sgt_spec = zeros(length(ks),length(tau));
for j = 1:length(tau)
   g = exp(-a*(t - tau(j)).^2);
   Sg = g.*S;
   Sgt = fft(Sg);
   l = fftshift(abs(Sgt));
   Sgt_spec(:,j) = l;
end

figure(6)
Sgt_spec = log(Sgt_spec + 1);
pcolor(tau,ks,Sgt_spec)
shading interp
set(gca,'ylim',[300 800],'Fontsize',16)
colormap(hot)
title('Sweet Child O Mine Spectogram')
xlabel('time (t)'), ylabel('frequency (f)')
```

Listing 1: fuction that creates first spectogram for GNR

```
clc; clear all;
figure(1)
[y, Fs] = audioread('Floyd.m4a');
trgnr = length(y)/Fs; % record time in seconds
plot((1:length(y))/Fs,y);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Sweet Child O Mine');
p8 = audioplayer(y,Fs);
S = y';
w = 10;
S = S(2*w*Fs:3*w*Fs-1);
%% Creating a spectrogram
L = length(S)/Fs ; n = length(S);
t2 = linspace(0,L,n+1); t = t2(1:n);
a = 500;
tau = 0:0.1:L;
k = (1/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);
Sgt_spec = zeros(length(ks),length(tau));
for j = 1:length(tau)
    g = exp(-a*(t - tau(j)).^2); % Window function
    Sg = g.*S;
    Sgt = fft(Sg);
    [C,I] = max(abs(Sgt(1:1800)));
    maxk = abs(k(I));
    n = 0.2;
    filter = exp(-n*(abs(k) - k(I)).^2);
    Sgtf = Sgt.*filter;

    l = Sgtf; % We don't want to scale it

    Sgt_spec(:,j) = fftshift(abs(l));
end

figure(6)
Sgt_spec = log(Sgt_spec + 1);
pcolor(tau+20,ks,Sgt_spec)
shading interp
set(gca,'ylim',[0 300],'Fontsize',16)
colormap(hot)
xlabel('time (t)'), ylabel('frequency (f)')
```

Listing 2: fuction that creates second spectogram for Pink Floyd's bass

```
figure(1)
[y, Fs] = audioread('Floyd.m4a');
trgnr = length(y)/Fs; % record time in seconds


p8 = audioplayer(y,Fs);

S = y';
St = fft(S);
L = length(S)/Fs ; n = length(S);
t2 = linspace(0,L,n+1); t = t2(1:n);
k = (1/L)*[0:n/2-1 -n/2:-1];

for j = 1:length(k)

    if(abs(k(j))>300)
        St(j) = 0;
    end
end

S = ifft(St);

p9 = audioplayer(S,Fs);
 playblocking(p9)
```

Listing 3: code that isolates the bass in Confortably Numb using low-pass filter

```
figure(1)
[y, Fs] = audioread('Floyd.m4a');
trgnr = length(y)/Fs; % record time in seconds
w = 10;

xlabel('Time [sec]'); ylabel('Amplitude');
title('Sweet Child O Mine');
p8 = audioplayer(y,Fs);

S = y';
S = S(1:w*Fs);
plot((1:length(S))/Fs,S);
L = length(S)/Fs ; n = length(S);
t2 = linspace(0,L,n+1); t = t2(1:n);
w = 10;

a = 500;
tau = 0:0.1:L;
a = 500;
yrecover = zeros(1,length(S));
table = zeros(length(S),length(tau));
Sgt_spec = table;
for j = 1:length(tau)
  g = exp(-a*(t - tau(j)).^2); % Window function
   Sg = g.*S;
   Sgt = fft(Sg);
  table(:,j) = Sgt;
  table(maxarray(j)-20:maxarray(j)+20,j) =zeros(41,1);
  kmax = k(maxarray(j));
  step = max(k)/(length(k)/2);
  I2 = round(maxarray(j) + kmax/step);
  I3 = round(maxarray(j) + 2*kmax/step);
  I4 = round(maxarray(j) + 3*kmax/step);
  table(I2-20:I2+20,j) =zeros(41,1);
  table(I3-20:I3+20,j) =zeros(41,1);
  table(I4-20:I4+20,j) =zeros(41,1);
end


for j = 1:length(k)

    if(abs(k(j))>2000||abs(k(j))<300)
        table(j,:) = zeros(1,101);
    end
end


for j = 1:length(tau)
   Sgt_spec(:,j) = fftshift(abs(table(:,j)));
end
Sgt_spec = abs(table);
figure(6)
Sgt_spec = log(Sgt_spec + 1);
pcolor(tau,ks,Sgt_spec)
shading interp
set(gca,'ylim',[300 800],'Fontsize',16)
colormap(hot)
  xlabel('time (t)'), ylabel('frequency (k)')
```


Listing 4: code that isolates the bass in Confortably Numb using low-pass filter